This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

An Inexpensive Upgradation of Legacy Cameras Using Software and Hardware Architecture for Monitoring and Tracking of Live Threats

Ume Habiba[1], Muhammad Awais[2], Milhan Khan[3], Abdul Jaleel[4]

[1]Post-graduate scholar, Department of computer science & engineering, University of Engineering & Technology, Lahore, Pakistan
[2]Assistant Professor, Department of computer science & engineering, University of Engineering & Technology, Lahore, Pakistan [3]Electrical
[3]PhD. scholar, Department of computer science & engineering, University of Engineering & Technology, Lahore, Pakistan
[4]Assistant Professor, Department of computer science, Rachna college of University of Engineering & Technology, Pakistan

Corresponding author: (e-mail: ume.habbiba@hotmail.com ; awais.hassan@uet.edu.pk).

ABSTRACT Surveillance through digital cameras is increasing exponentially. A majority of these cameras are not smart cameras; therefore, they send their video stream to a central server where it is processed and analyzed for any threats. Typically, human operators or machine learning algorithms at cloud, analyzed and processed the post-event videos to track and locate the perpetrator or victim. The centralized approach leads to two primary shortcomings: 1) the high cost of cloud infrastructure; 2) lack of instant tracking and detection of the threat. One solution is to replace these legacy cameras with the smart cameras so they can process information locally. Although the solution is costly, it could solve the real time threat detection issues. However, the need for a central server remains there, to construct the path of threat, when threat moves from one camera view to another. The existing distributed architectures for threat tracking, shifts the load of threat capturing and processing from central server to the edge nodes, which in turn reduces the computational power but do not remove the role of central server completely. These architectures doesn't equip each camera of processing and communicating with each other. Further, in the existing distributed architectures the local cameras are not able to store the path of the threat individually, and transmits the captured trajectory to the central body. This research proposed a second alternative that makes use of legacy cameras through additional hardware and software components such that they can process information and collaborate locally. The research addresses the challenge by introducing a low cost distributed threat tracking framework that allows the single camera to identify the threat and communicate its information to other cameras without involving the central server. The framework stores the information in a lightweight architecture that is inspired by the blockchain storage algorithm. The system also allows querying the path traveled by the threat at any stage. To evaluate the system, we performed two simulated experiments: one with a central server and another with the proposed distributed system. The results of the experiments showed that the time to track the threat through the proposed system was lower than the existing centralized system. Moreover, the proposed system predicted the paths of threats with an accuracy of 85.49%. In the future, the technique may be improved with reinforcement learning and other machine learning techniques.

INDEX TERMS Video Surveillance, Smart Cameras, distributed computing, software architecture, Peer-to-peer computing

I. INTRODUCTION

April 15, 2013, is one of the darkest days in the history of the United States of America as it witnessed two disastrous explosions. These explosions killed more than three people, while hundreds of others received severe injuries. Later, the two suspects of the bombing were captured by sifting through CCTV videos of several hours [1]. The increased terrorist and crime incidents have alarmed nations around the globe and necessitated proper surveillance in cities [2].
Different sensors are being used these days for surveillance and especially path tracking of threats. These sensors include binary cameras [3], depth data [4], and digital cameras [5]. The digital cameras are most important source of path tracking due to their high resolution captured images, and low cost compared to other sensors.
A large number of cameras, for surveillance, have been installed in all big cities of world by their governments. These devices their continuously capture the video stream of the area under their observation [6]. Usually, these cameras feed their video stream to cloud infrastructure [7]–[9] where it is stored, processed and analyzed [10]. To track the path of any victim or perpetrator, these videos are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

processed by the machine learning algorithms [11]–[13] and the human operators.

The existing methods, in literature, for the surveillance have many drawbacks. For example, a study [14] has revealed that the monitoring of multiple camera views puts a significant burden on the human operator that results in time consumption, exhausted eyes, and error-prone work [15]. The other methods [11]–[13] those employ intelligent learning algorithms for the detection and re-identification of persons do not offer collaboration and communication between cameras. However, these intelligent surveillance systems are unable to query about specific threat information from their neighboring cameras and fails to live track the threats. Other solutions uses the central server where the information is processed but these centralize systems have three primary challenges: 1) the high cost of cloud infrastructure; 2) real time monitoring of threat; and 3) tracking the path of the object. There is a need for a distributed system where nodes can query about threats from neighbors and perform live tracking of a threat without the involvement of a central server.

The goal of this research is to propose a software architecture that enables legacy cameras, without an expensive cloud infrastructure, to establish ad-hoc networks automatically without any human intervention, identify the threat, transmit and seek information of the threat from neighboring cameras, track the threat movement and allow to query threat information from the distributed network whenever it is required. We targeted following research questions to achieve our goal.

1) How do you enable legacy cameras so they can establish a distributed network automatically without the intervention of humans or a centralized system?
2) How can these modified cameras track the threat from archived videos without any time expensive machine learning algorithms or human intervention?
3) How can threat information be stored, co-related, and queried from these distributed camera nodes?

Our contribution is that we enabled the legacy camera with the help of a low-cost hardware to communicate with other cameras and detect the threat locally. Once the local camera detects the threat, it stores the relevant threat info (such as timestamp when the threat is detected and reference frames), predicts the threat direction and informs the neighboring cameras about the threat so they can track it further. In this way, only the relevant threat information is required to be saved, and the upgraded cameras track the path that can be accessed later.

II. Literature Survey

Body tracking, and detection are quite useful in multiple fields including robotics [16]–[18], health related issues [19], and industrial academic areas [20] but most of the applications are in surveillance. Surveillance of a human body can be done by different methods in the existing literature [21]–[26]. For identification of a threat the feature extraction, classification, and face recognition are the most important steps and the technique used for these steps include: artificial neural network, infrared sensors, and human segmentation [27]–[30]. These researches tracks the human body by re-identifying it in different cameras, or by action recognition made by different body parts of a human.

Chen et al. proposed "City Eyes", a cloud-based computational framework for developing intelligent surveillance applications [7]. The authors integrated “City Eyes” in multiple surveillance systems of different cities and showed a reduced time in continuous monitoring of surveillance videos. In order to fulfill the QoS requirement and to optimize the allocation of VM resources, Hossain et al. presented a resource allocation scheme [8]. The scheme streamed composite media in a cloud-based video surveillance environment. The elastic cloud-based platform stored all video streams that were captured and transferred by the surveillance cameras. Li et al. investigated the processing of massive floating car data (FCD) for traffic surveillance in cloud computing environments [9]. Empirical studies showed the potential of cloud computing for providing various solutions for on-demand geospatial data-intensive applications.

Shao et al. developed an intelligent system with smart front-end cameras for surveillance and pre-alarming [31]. The smart cameras were able to pre-alarm and store any unusual event in the database. Chandana et al. [11] proposed a surveillance system using “thing speak” and raspberry pi. The raspberry pi enabled the cameras to capture the image and detect the motion of a person. The images of individuals were captured only after the detection of motion signals which in turn reduced the power consumption compared to the surveillance system that continuously captured the videos. Abas et al. developed “SlugCam”: an outdoor wireless smart camera network [12] where nodes were intelligent enough to change their monitoring behavior when the passive infrared sensor (PIR) detected any motion. Wang et al. demonstrated a paradigm of "tweeting" cameras [13]. The software architecture of tweeting cameras was able to recognize, detect abnormal events through Sony IMX219 8-megapixel sensor. The camera also tweeted about exciting events on social media and received replies from humans for the learning process.

For processing and monitoring, Zhang et al. sent data captured by cameras to cloud storage [32]. This data transmission results in high response latency and bandwidth constraints, which in turn proves this solution to be inefficient and expensive. To overcome the constraints of cloud storage, the researchers, afterward tracked and identified threats through intelligent single and multi-cameras. Tang et al. tracked the path of a person by re-identifying him in a monocular video of the crowded scene [33]. The authors used a novel graph-based approach for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

linking and clustering the track of a person. Results showed that this method outperformed the existing benchmark. Beyer et al. integrated re-identification (ReID) with multi-target multi-camera tracking [34]. This integration resulted in an optimal Bayes filter. This filter avoided the requirement of data association and dependency on boundary boxes for tracking.

For detecting and tracking people in-depth images that were captured through a time-of-flight camera, Stahischmidt proposed a method and an application [35]. The detected persons were tracked by a Kalman filter, and their adopted trajectories were stored. Images captured by the cameras were taken perpendicularly from the top-right angle and were stored locally in the camera. The solution was centralized as other cameras were not able to process the information independently. Tesfaye et al. proposed a three-layered hierarchal approach for tracking people in multiple non-overlapping cameras [36]. This method took a video and set of detections as input and performed within-camera and across camera tracking. The camera communication was not allowed, and the input was not taken directly from the cameras.

Liem & Gavrila proposed a multi-people position tracking algorithm for overlapping cameras [37]. The similarity of a person and running track were mapped using a set of hints such as motion style and appearance of the person. The authors detected foreground maps by using background subtraction method. The tracking was done by associating these detections to the previously tracked individuals. The experiments showed that this system outperformed with multi-person datasets having overlapping cameras, and track consistency were also improved. This system did not allow performing any query about a specific threat.

Bhuvana proposed an object tracking algorithm for targeting the bandwidth and energy limitations in the information exchanged among surveillance cameras [38]. This method restricted the number of cameras participating in the information sharing process. The surprise selection method enabled the cameras to decide whether their information was essential or not. This method showed improved tracking accuracy.

Wang et al. proposed a surveillance system that enabled communication of cameras with edge nodes and reduced the computation delay on central servers [39]. The delays were reduced because computation and storage resources shifted from the centralized data center to the edge nodes. These experiments showed that the system was more rapid, responsive, and flexible. However, the cameras in this network were not enabled with inter-camera communication. Jiang et al. proposed a person re-identification framework based on the orientation of a person [40]. Cameras in the framework were enabled with inter-camera information exchange i.e., the camera was able to share information with its different modules but not with other cameras. The camera in the framework shared images of the same person based on his discriminative appearance features for associating inter-camera trajectory and achieving inter-camera Spatio-temporal constraints. The communication of camera with different modules helped in person re-identification despite occlusions.

Kumar et al. developed a person re-identification algorithm with distributing computing capabilities in non-overlapping camera networks [41]. The cameras in the system were able to self-process the threats and pass relevant information to a primary camera. The primary camera was able to query about a particular person from the neighborhood cameras. Well, the trajectory of the identified or tracked persons was not stored in the system. For detecting abnormal events in frames, Wang et al. proposed an algorithm for distributed cameras network [42]. The authors presented a multi-kernel strategy for benefitting from the different views captured by multiple cameras. Although this work identified malicious activity, but the cameras involved in the network were not able to automatically configure and track or store the path of a threat.

The most of available solutions provide surveillance through a single camera and with the involvement of cloud infrastructure. However, a very low number of researches existed that deal with multiple cameras for threat detection and tracking.

III. Proposed Method

For video surveillance, the proposed distributed network consists of multiple cameras and each camera is called a node. To convert the node into a smart node, we proposed a software architecture (Figure 1) that can run on the raspberry Pi. Raspberry camera was provided in Camera Serial Interface (CSI) of all other cameras which in-turn formed a network. The distributed network consists of four major components: network manager, threat handler, path generator, and controller.

Network manager is responsible for storage of all nodes and prediction of next location of the threat. Records required by network manager are stored in “camera lookup” log. Threat handler component stores information about all threats and uses the convolution neural network (CNN) for threat identification. In case of a threat, the information about the threat is communicated to the neighbor nodes. The path generator component produces the path against the query about a threat. The controller is the major component that is responsible for the communication among the nodes and the addition of nodes in the network. The nodes in the network communicate with each other by sending messages in the form of packets. The following section provides the details about these components.

A. Network Manager

A legacy surveillance camera can track the threat up to a limited range because of its finite and fixed position [25]-[26]. Once a camera has identified the threat, it starts

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

monitoring the threat, and when the threat is moving out to its field of view, the camera has to inform neighboring cameras that the threat is entering into your viewing area. In order to achieve the functionality, each camera should be aware of its immediate neighbors, Fig 3. There are two options one is to configure each camera manually, and a human technician provides the information of neighboring camera. However, on each node, it requires to add information of all neighboring cameras manually that is a massive task if new cameras keep coming up, the position of some cameras is changed, or they stop working.

The most important information is the coverage area and view frustum of each camera. Each camera knows its fixed location, its field of view (FoV), and coverage distance. When a camera gets online, it broadcasts these parameters to other cameras. From this tuple, C (location, FoV, coverage distance), other cameras calculate the two more points B (Eq. 1) and D (Eq. 2) for each camera such that A, B, D make a triangle of its coverage area Fig 3 (a), Fig 3 (b). The blind spot between two cameras where no coverage is available is also shown in Fig 3. Also receiving the camera calculates the relative location of another camera regarding its position Fig 4. These relative positions could be from one of four quadrants Fig 4. using the function given in Eq. 3; this relative location is used by Algorithm 2 while invoking the neighboring cameras for a potential threat. Table 1. shows a comparison of the proposed system having with the existing solutions for threat detection and tracking.

TABLE 1: COMPARISON OF PROPOSED WORK WITH EXISTING SURVEILLANCE SYSTEMS

	Threat path identification or tracking	Automatic camera management	Tracked path storage	Information Sharing among cameras	Hybrid system for identifying, tracking, path storage
Liem [9]	(Automatic) using the 3D reconstruction of the scene for detection & tracking	Manual	local	None	×
Stahischmidt et al. [7]	Automatic (Kalman-based multi-object tracking)	Manual	local	none	×
Jiang [12]	Automatic through deep learning architecture	Manual	local	None	×
Wang [11]	through elastic dynamically launched Virtualized Network Functions (VNFs) on edge servers	Manual	Edge cloud storage	With edge nodes	×
Chen [21]	PaaS controller (Video Analysis Platform-as-a-Service)	Manual	Local	Cloud	×
Proposed Work	Through Raspberry pi, and distributed classifier	Automatic	local	inter-camera information sharing	✓

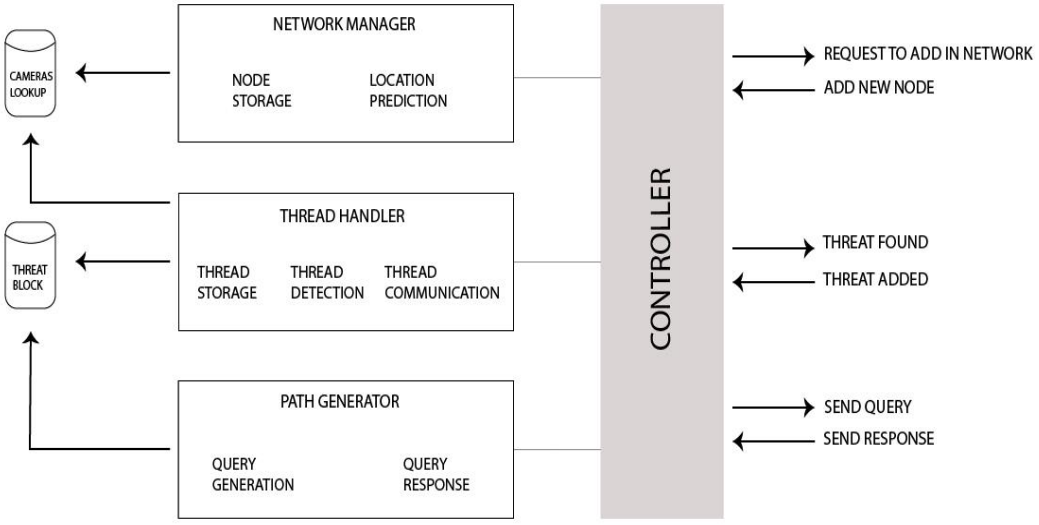

Figure 1: Overall system architecture and services

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

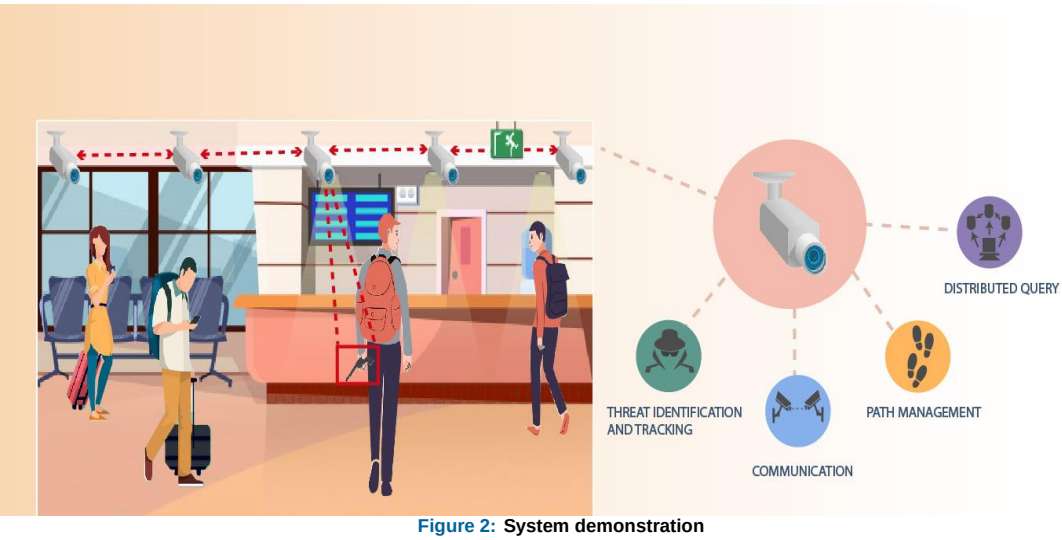

Figure 2: System demonstration

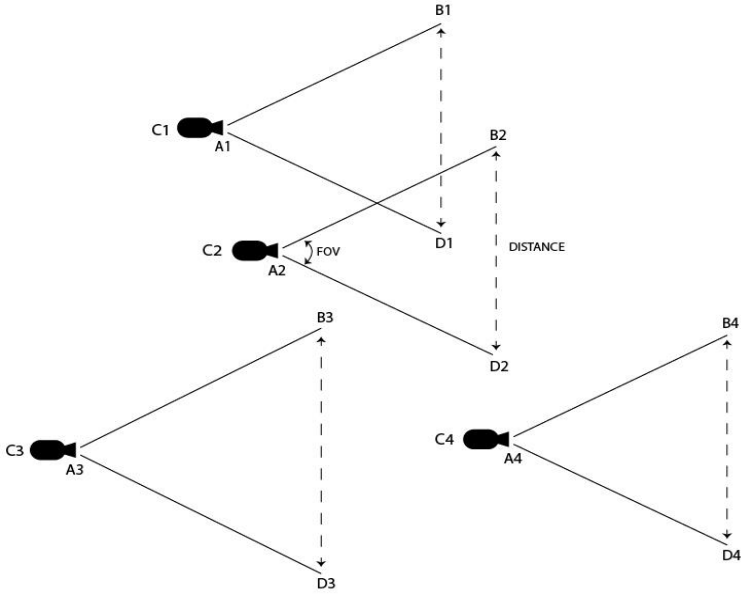

Figure 3 (a): Neighboring camera networks

$$B = \frac{FOV * f}{h} \qquad (1)$$

Here, 'B' is the distance of a camera C_i from its left neighbor. 'FOV' is the field of view of the camera as specified by the camera designer. h is the height of the camera sensor, and f is the focal length of the camera lens.

$$D = \frac{FOV * f}{h} \qquad (2)$$

Here, 'D' is the distance of a camera C_i from its right neighbor. FOV is the field of view of the camera as specified by the camera designer. 'h' is the height of the camera sensor. For calculating the relative position of a camera C_1 with respect to other camera C_2 at first quadrants of both cameras are calculated. We proposed a local storage table that stores all the information of the neighboring cameras Table 2.

Table 2: Lookup table of C_2

Node ID	Angle	Distance (meters)	A	B	D	Direction
C_2	232	20	20,15	30,25	18,25	θ3
C_4	180	10	18,20	25, 12	10,20	θ4
C_3	210	60	10,18	12,24	20,25	θ1

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

Suppose P_{dir} is the predicted direction of the camera. Let Θ_i be the quadrant angle of the current camera (C_i) that has captured the threat. Let Θ_k, Θ_{k+1}, Θ_{k+2},,,,,,, be the angles of all neighbors of C_i. The difference between the angles of the current camera and the angle of all other neighbor camera is calculated. The camera with a minimum difference is regarded as a neighbor of C_i and is sent information about the threat and its predicted direction.

$$\text{Min } \{(\Theta_i - \Theta_k)\ (\Theta_i - \Theta_{k+1}) \ldots..\ (\Theta_i - \Theta_n)] = \text{Neighbor} \quad (3)$$

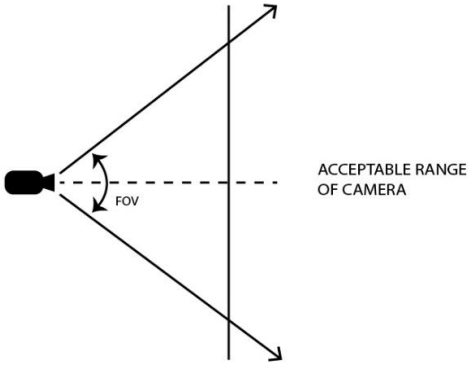

Figure 3 (b): **Neighboring camera networks**

Fig 4. shows the arrangement of cameras in a plane and the way with which relative location of a camera is determined

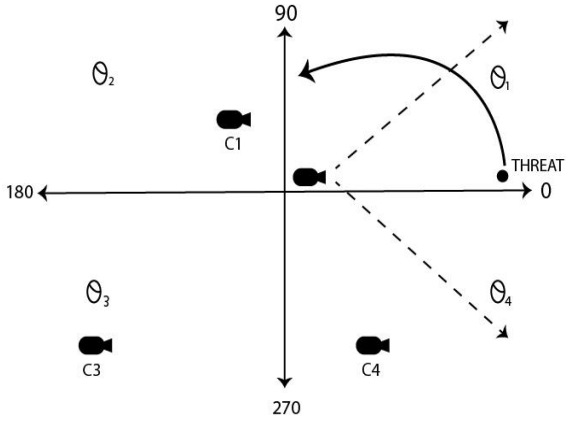

Figure 4: **Relative location of a camera**

Algorithm 1: Node Manager

Input: Node N
Output: Acknowledgement (Sender Info)
Flag=false

foreach (Old_Node in lookupTable.Entries) **do**

if (N->IP = Old_Node -> IP **) then**
flag= True //node already added
sendAcknowledgement(Old_Node)
endif

endfor

if (not flag)
pointB = (fov * f) / h
pointD = (fov * f) / h
quadrant= FindQuardrant (Longitude, Latitude)
end if
Entry = CreateLookupTableEntry
(N.ID,N.angleBetweenCamera,
N.distance,N.location,pointA,
pointB,pointB,Quardent**)**
lookupTable.Add (entry)
return acknowledgement(entry)

Threat Handler

This module consists of three sub-modules: Threat Detection, Threat Communication, and Threat Storage. First, we discuss these sub-modules, and then we give a complete algorithm for threat handling process.

i) Threat detection

The proposed system consists of different camera nodes those are connected to form a distributed network. Each camera captures and monitors the scene in its range. The camera node consists of a Raspberry Pi that acts as a tiny processor and serves as a host for videos processing, storage, identification of threats, and further communication and warning to the neighbors.

Threat detection and identification at early stages prevent severe losses. In order to save the time and to avoid the sending of all video streams to the server, the proposed system uses this local module to identify the threat. This module identifies threats from a video sequence that is captured by a camera node in the network. Each image/frame captured by a camera node is first preprocessed by pre-processor before threat detection or identification. The preprocessor follows some steps for preparing image for threat identification. These steps are mentioned in Fig 5.

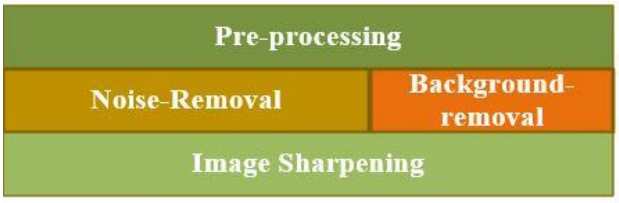

Figure 5: **Block diagram of preprocessing procedure**

Step 1: (Noise removal): Noise is introduced in an image at the time of its acquisition and it is introduced due to different factors including inappropriate light settings, corrupt image sensor, and dust particles on the camera. We used the order statistical filter to remove noise in the captured images by the camera [45]. It is a nonlinear filter whose response depends on ordering of pixels encompassed by the filter area.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

Step 2: (Background Removal): For background removal from the frames, we used background subtraction technique using concept of running average [46]. In this method, video frames were analyzed. A comparison of running average of current and previous frames, provides background and foreground model. The foreground model is extracted from the comparison, by detecting the active objects.

Step 3: (Image Sharpening): For sharpening the images with removed background, we used un-sharp mask [47]. This technique first uses a blurred version of the original image. This blurred version is then subtracted from the original image. This subtraction points out the presence of the edges, hence creating the un-sharp mask. An increased along the edges using the above mask, the product is a sharpened image.

For proper identification of the threat, there is a need for visual features' extraction from the video sequence. The essential features are learned and extracted using CNN.

CNN models require training on a huge amount of data for learning before any usage. This training causes high time consumption, and to avoid this; we have used a pre-trained CNN model. Fig 6. shows threat identification module of the proposed system. For recognizing threats' face, we used visual geometry group (VGG) Face-16 CNN [48]. Structure of VGG Face-16 has 13 convolutional layers, five pooling layers, and three fully-connected layers. VGG Face-16 was trained using a publically available dataset [49]. After the convolution operation, CNN produces feature maps. The size of these feature maps is determined from the width or height of the filter, the width or height of the input image (or feature map) before it enters the convolutional layer, the amount of padding in the convolutional layer, and the number of strides [50]. Rectified linear unit (ReLU) layer follows the convolution layer, which used polling windows for reducing dimensions of obtained features and smoothing the features extraction process. For window sliding Partial Least Square method (PLS) is used and the features used by Robson et al. [51]. The final layer, fully connected layer (FCL) consists of a softmax function for normalizing the inputs and yielding categorical distribution of each class function.

ii) Threat Communication and Storage.

If the upper layer detects a person as a threat, it predicts the exit quadrant where the threat is heading. The quadrants are defined with reference to the point of the camera. The module uses the direction of the head in the captured set of images. This method takes a face as a parameter and returns its direction using the corner points of facial features (head, nose, eyes). The location of the camera is required to predict the quadrant coordinate. We used the method given by Xing et al. to find the coordinates of the threat [52]. Once the coordinate has been found, the quadrant is determined by the following function:

```
FindQuadrant (longitude, latitude)
{
  if (longitude > 0  && latitude > 0) { quadrant = 1}
  elseif (longitude < 0  && latitude > 0) { quadrant = 2}
  elseif (longitude < 0  && latitude < 0) { quadrant = 3}

  elseif (longitude > 0  && latitude < 0) { quadrant = 4}
}
```

After predicting the threat of future quadrant, the camera creates a block with threat hash, its-IP, predicted quadrant, predicted direction, current block id, and nearest neighbors in the predicted direction. This block-chain based module stores the path history of a particular threat. Whenever a camera node detects a threat, it creates a block for the threat that consists of threat_id, time, and date. Each block consists of a hash value, which acts as its unique identifier. The time, date (when the camera identified the threat), and the block_id of the notifying node play a vital role in the calculation of the hash value. The value of the block is empty for the first camera that detects the threat; in the blockchain, this block is termed as "genesis block." The block does not have next camera IP it remains blank until the threat enters into the specific camera. After generating the genesis block, the system logged information in the local database and forwarded the block to all registered cameras (present in a lookup table). Now, all the relevant cameras have the genesis block for the threat. When a camera detects a threat, it broadcast the information to cameras so they can discard the genesis block and it also informs the parent camera. The parent block appends the IP of receiving camera into the block so it can help to create a chain for the threat. Now, the second camera repeats the process, but this time it creates its block and leaves the next camera attribute blank. Once the threat leaves the camera, it sends the block information to another camera of the predicted quadrant.

Algorithm 2: Threat Handler

```
Input: VideoFrame frame
pFrame = preprocess (frame)
featureMap = GetFeatures(pFrame)
isThreat = Model.Classify(featureMap)

 if (not isThreat) return
 else
 Direction = PredictThreatDiretion (pFrame)
 Position=PredictThreatPosition(PFrame)
 Quadrant = FindQuadrant (longitude, latitude)
 NewHash=generateHash(Node.IP,threat)
 Old_Block =threatBlocks.find(pFrame)  //if threat is
            already communicated by any other camera
 ParentBlock = GensisBlock
     if  Old_Block !=Null //Threat BlockAlready Exist
     Old_Block.NextBlockHash=newHash
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

SendAcknowlegment(Old_Block.senderIP,OldBlock.HashID,newHash)
ParentBlock= Old_Block.HashID
End if

B. Threat Path Generator

After threat identification, the node stores the information of threat into the local database in the form of a block. However, the block does not store the path of the threat; the block only contains the location of the next block and its Hash ID.

For path construction, the node queries the path of a specific threat from another node where the threat has moved. For the purpose, a distributed query can be passed using the BlockID that is to be tracked, IP address of the current node from where the query is to be made and previous path of the threat if any. A node N can broadcast a query to its neighbor node, about a specific threat having blockID. The neighbor checks the log for BlockID. If a neighbor finds a threat in its log, it appends its location into the path and checks whether the next block is null or not. If it is not null, it calls the function construct path for the next camera. If it is null, the sequence of blocks is returned to the inquiring camera.

PreProcess(FindImage)

Input: QueryStarter,Sender,BlockID,Path
Output: ForwardQuery/SendResults
Found=false
For each **B** in Blocks
If (B.ParentBlock==BlockID)
Found=true
Path.Append(NodeID,B.BlockID)

SendQuery(B.NextNodeID,B.NextBlockID)
End if

If (Not Found)
SendResult (OueryStarter,Path)
End if
End For

Algorithm 4a: Path Generator

Method Generate Threat Path Request

Input: Preprocessed-Image
Output:
RBlock=null
For Block B in threatBlocks
If B.frame == Preprocessed_Image
Rblock=B
End if
End For
If Rblock !=null
Path.append(Current.NodeID,RBlock.BlockID)

Path=SendQuery(RBlock.NextNodeID,RBlock.NextBlockID,Path)
Wait For Response

Print Path.
End If

Fig 7. Shows how the path of a particular threat is stored in each camera installed in the way where the threat has passed. For example, in order to query about a particular threat (John), the inquiring camera sends the BlockID to its neighbor cameras. If the neighbor cameras do not find the threat information, they forward the request further to their neighbors. In another case, the inquired camera will go for the current ID of the threat. The current id contains information about the cameras by which the threat was being captured plus the id of latest camera that has spotted the threat.

C. Controller:

Each message created in the proposed system is exchanged with neighbor cameras through the controller. This component uses the services of Raspberry Pi. Raspberry Pi is attached to each monitoring camera, provides a dedicated socket for continuously listening to the requests from its neighbors. Requests from the neighbors can be one of the three types: threat alert, a new node broadcast, and distributed query. Whenever a message about a particular threat is received from the neighbor camera, the communication layer redirects the message to threat handler module of the current camera. Also, when a node is injected, it broadcasts "add" request to all nodes in the network. The listening socket on Raspberry Pi of receiving nodes forwards the request to its node manager for the addition of the new node into the routing table. In case the request is of the distributed query, the listening service forwards it to the path manager.

Algorithm 5: Controller

1: **Input:** Req_Service

2: **if** (Service = add or remove node (N)) **then**
3: NetworkManager (N)
4: **if** (Service = Monitor_Threat (**Threat_ID**))
5: ThreatHandler (Threat_ID)

6: **if** (Service = Query (Threat_ID)) **then**
7: PathGenerator (Threat_ID)

8: **endif**
9: **endif**
10: endif

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

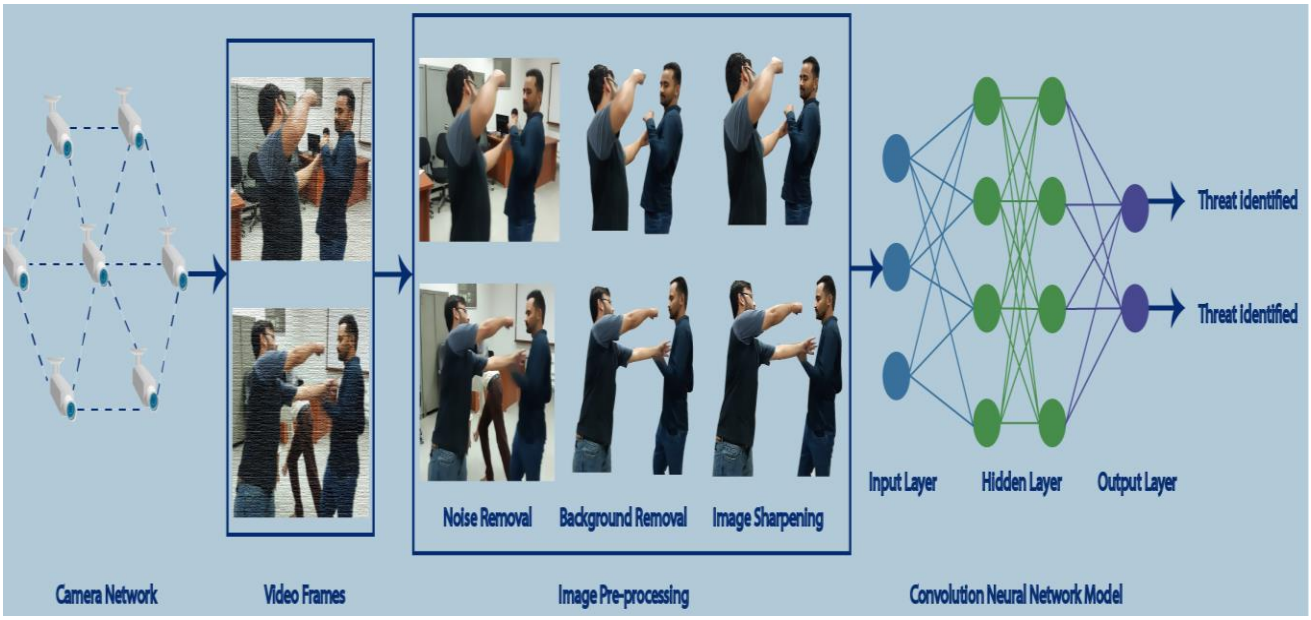

Figure 6: Convolution neural network layer

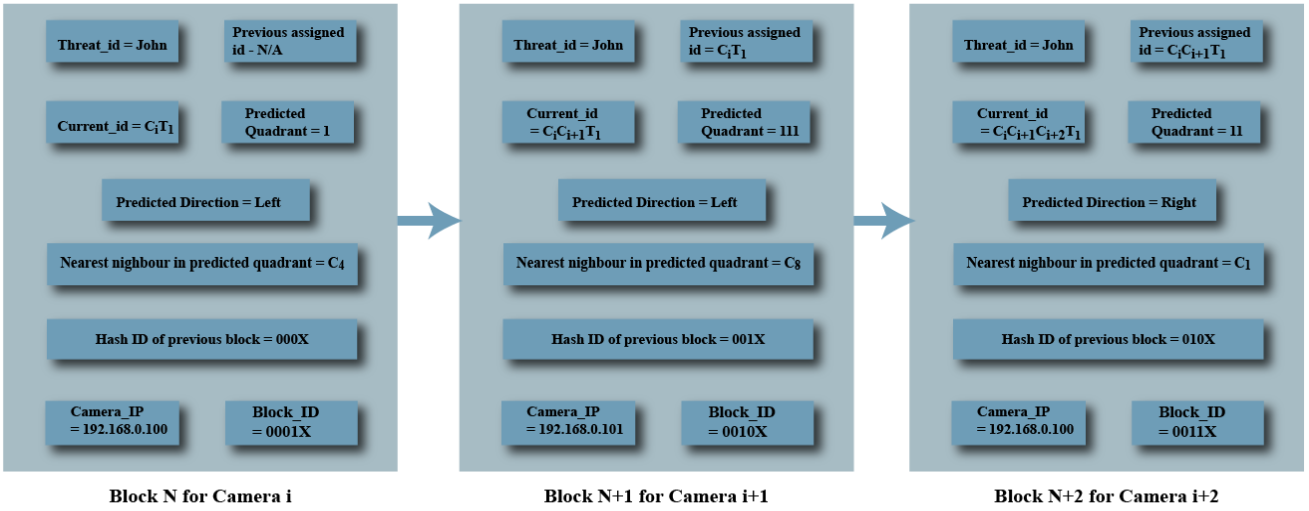

Figure 7: Block chain demonstration of threat information

IV. Experimentation

A. Experimental Setup

Our experimental setup consisted of the 35 surveillance cameras installed at various locations of a university campus. We used raspberry Pi 3B+ with Raspbian Jessie OS booted from a 64G microSD card as in intelligent module to be attached to each camera. The cameras in this setup were installed at the incoming and exit points of eight departments of the university. In total, a hundred video clips were passed to the Raspberry module in which one hundred and fifteen threats were explicitly simulated. These threats were people who performed abnormal and ambiguous activities. The dataset used for classifying threats through classifier on central server and the one running on Raspberry Pi is “Behave” [53]. This dataset consists of view of people acting out in different situations. The data is captured at 25 frames per second. The resolution is 640x480. The videos clips are provided in the dataset that can be classified as frames. These frames are available as a set of JPEG single image files. The normal behavior of the person are labelled as meet, walk together, split, and ignore. While the abnormal behavior is labeled as: fight, and chase.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

B. Experiment I: Threat identification through a centralized network of cameras

The first experiment was performed to assess the time taken in identifying a threat by a centralized system. In this experiment, the hundred multiple video clips, collected through installed cameras in the experimental environment, were fed to the central server for the analysis. A classifier running on a central server identified threats was trained on multiple datasets [53] of videos containing normal and abnormal behavior of the people. The same classifier was also used for the distributed system in experiment II. Moreover, the communication among the camera nodes was not allowed in this first experiment.

C. Experiment II: Threat Identification through proposed system

The second experiment was performed to analyze the time taken to identify a threat via proposed distributed system and the accuracy of the path prediction. In this experiment, the proposed framework was used to track and identify the threats. Threats were identified by services provided by Raspberry module attached with each camera node. The classifier on Raspberry pi was trained on multiple datasets [53] of videos containing normal and abnormal behavior of the people. After detection of a threat, the camera predicted and forwarded threat's face to the relevant nearest neighbor for live tracking of the threat. Message passing among camera nodes was in the form of packets and the number of packets received by the receiver node were logged and compared with the number of original packets sent.

Also, three nodes were added in to the distributed network at location L_1, L_2, L_3 respectively, where the configuration of each camera was updated automatically. The added nodes sent requests, containing their IP and location (longitude, latitude), to the neighbor nodes. The neighbor nodes were automatically configured to add the received IP in their routing table.

V. Results

VGG 16 used for threat detection was classified using the dataset "Behave" [50], and the abnormal activities e.g. chase and fight were detected using this dataset. VGG 16 achieved 96% accuracy for classifying the threats in above dataset. These results were compared with those of HMM [5], SVM [54], HMM based GMM [55], and the comparison is listed down in Table 3.

Table 3: Comparison of VGG-16 with existing techniques

Technique Applied	Classification accuracy achieved
VGG-16	96.09%
HMM based GMM	84%
SVM	94.9
HMM using depth Silhouettes Context features	83.92

Table 4. lists down the processing time to identify 115 threats in experiment 1 and experiment 2. Path tracked by the proposed system for a specific threat was matched with the original path followed by a threat which was already known and stored in the system. Table 5. lists down paths predicted by the proposed system for a few sample threats and with the original paths adopted by the threats. The locations in the university campus, where a camera node is installed, is named as L_1, L_2, L_3,... L_N.

Table 4: Sample Threat tracking time for both experiments

Time is taken to identify different threats in the experiment I	Time is taken to identify different threats in experiment II
1302 seconds	1000 seconds
1477	1112
1132	899
1298	990

Table 5: Sample paths of threats predicted by the proposed system

	Location	Path Predicted	Original Path	Is-Accurate
Threat 1	L1	A-B-D-L-G	A-B-D-L-H	No
Threat 2	L2	A-B-C-L-H	A-B-C-L-H	Yes
Threat 3	L3	B-C-D-L-G	B-C-D-L-G	Yes
Threat 4	L4	F-G-H-E-F	F-G-G-E-E	No

The system predicted paths of 85.49% threats precisely the same as their original path.

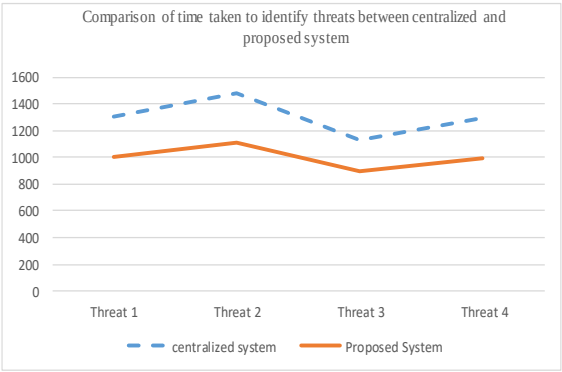

Figure 8: Graphical comparison of threat identification time of centralized and proposed system

Figure 8. shows a graphical comparison of the time taken by the centralized and proposed system to identify random threats.

Table 6. shows a comparison between the routing tables updated manually by the human operator in case of addition of three nodes at location L5, L7, L2,.and routing tables updated automatically in case of the same addition.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2964778, IEEE Access

Table 6: Manually updated routing table vs automatically updated routing tables

The manually updated routing table						The automatically updated routing table					
Dest	*Area*	*Hop Cost*	*Next Hops*	*Network address*	*interface*	*Dest*	*Area*	*Hop Cost*	*Next Hops*	*Network address*	*Interface*
N_1	1	4	RT_1	10.0.10.0	Eth1	N_1	1	4	RT_1	10.0.10.0	Eth1
N_2	1	4	RT_4	255.0.100.1	Eth0	N_2	1	4	RT_4	255.0.100.1	Eth0
N_3	0	3	RT_3	192.168.0.100	Eth2	N_3	0	3	RT_3	192.168.0.100	Eth2
N_4	0	1	RT_2	192.168.10.1	Eth1	N_1	0	0	RT_2	192.168.10.1	Eth1

The error rate of the automatically updated routing table through the proposed system was 0.012%. The average time is taken by a camera to update its routing tables automatically was reduced to 4.5 seconds then the human operator who took an average of 8 minutes to update the routing table of a camera.

VI. Discussion

In this research, we have proposed low-cost system architecture for identifying and tracking threats through a distributed network of video surveillance cameras and IoT sensors. A Raspberry Pi module was attached to each camera node which offered various services through the software modules. All cameras were connected in a distributed network, and the proposed architecture enabled these cameras to identify and track the threats locally. At first, when a camera node identified a threat, it assigned an identification number to the threat. Then, the camera node predicted the next direction of the threat. Data about the identified threat is forwarded to the neighbor node in the same direction where the threat is moving. Communication among nodes helped neighbor cameras to track the path of threat using threat and path management services of Raspberry Pi.

The proposed solution offers a service that allows the cameras to automatically form a distributed network without any human assistance (experiment II). The node manager module was responsible for updating the routing tables of each node in case of an addition of a new camera in the network. When a node is added into the system, it broadcast the add request to other cameras in the network. After receiving the request, the nodes in the neighbor update their routing tables for any matching node with the same IP as of the requested node. The routing table stores the location (longitude, latitude), direction, and IP address of neighbors. The routing tables of all devices were updated when three new nodes were added into the distributed network at the location L5, L7, L2. The manually updated tables were compared with the automatically updated routing tables; the later showed an error rate of 0.012%. This dynamic insertion of nodes reduced the extra effort for manually updating the routing table associated with each camera from 8 minutes to 4.5 seconds. However, the removal of a node was not available in the system.

Raspberry Pi on each camera in the distributed network enabled local threat identification and tracking. The future direction of each threat was predicted and the current information about the threat was stored in the local storage of the camera. The proposed system predicted paths adopted by the threats (Experiment II) with an accuracy of 85.49% as of the actual paths of the threats (Table 5). The high accuracy of the proposed system was due to the local monitoring and threat information storage by each camera rather than a centralized unit.

Threat information about each threat was saved in the local storage associated with each camera along with the threat id. Information about each threat can be queried through each camera. A query about a specific threat was forwarded to the nodes in the direction where the threat was moving. A camera that identified the threat logged the data in the system, predicted the next direction of the threat and forwarded the query to the neighboring node in the same direction. The same process continued until the threat was not available in the frame of any camera in the network. Distributed query, reduced the average time taken to identify a threat (Table 4) as the identification was made through distributed nodes (experiment II) rather than the central system (experiment I).

The proposed system has reduced the overall time cost for identifying and tracking the path of the threat. It has also reduced the need and cost of data transmission and receiving from/to the centralized server. Blockchain based data logging about each threat, and the processing capability of each node has reduced the overall cost of data submission and retrieval from a centralized server.

VII. Conclusion

Based on the discussion above, we can conclude that instead of replacing hundreds of thousand cameras, the capabilities of these cameras should be enhanced with the proposed software architecture that can run on low-cost smart hardware. The proposed software architecture and sensors give cameras the capability to communicate with neighboring cameras, identify and track the threats without the need for costly cloud-based infrastructure. Moreover, we came to the conclusion that a distributed system is better than a centralized system due to time and cost factors. This strongly adheres to the results of the experiment. The proposed system also comes up with some limitations i.e. it does not consider the fast rotations in the captured images. If a video frame contains immediate rotation of faces, the proposed system will fail to identify them. The proposed

system does not provide coverage / surveillance in blind spots. The network in the proposed system does not provide continuous coverage i.e. if a threat is recognized first by a camera, the next time the threat appears in the same camera, it will not be recognized by this camera. The study can be extended to reduce the error rate in path prediction of the threat together with passing an alert message concerning authorities after identifying a threat.

Miss. Ume Habiba is a post Graduate student of computer science at Computer Science and Engineering Department at the University of Engineering and Technology, Lahore, Pakistan. She is also working as a research assistant in UET Lahore. Her research interests include Gamification and adaptive learning.

Dr. Muhammad Awais Awais is Gold Medalist of Punjab University in MCS computer science. He has completed his PhD. Computer Science from University of Engineering & Technology, Lahore, Pakistan. He is currently working as an assistant professor at Computer Science and Engineering Department at the University of Engineering and Technology. His research interest includes Artificial Intelligence, Reinforcement Learning, Adaptive eLearning Systems, and Affective Computing.

Muhammad Milhan Afzal is serving as a Lecturer in the Computer Science department of the University of Agriculture, Faisalabad, Pakistan and a Ph.D. scholar at University of Engineering and Technology Lahore, Pakistan. His research interests are Blockchain, Software Metric, and Software Design.

Dr. Abdul jaleel received the B.S. degree in Computer Science and Engineering from UET, 2006. He completed M.S. in Computer Science in 2010 and then received Ph.D. degree in Computer Science from the same university in 2019. He is working as Assistant Professor at the same college. His research interest includes the development of self-managing software applications.

References:

[1] J. C. Klontz and A. K. Jain, "A case study on unconstrained facial recognition using the boston marathon bombings suspects," *Michigan State Univ. Tech. Rep*, vol. 119, no. 120, p. 1, 2013.

[2] S. Ojha and S. Sakhare, "Image processing techniques for object tracking in video surveillance-A survey," in *2015 International Conference on Pervasive Computing (ICPC)*, 2015, pp. 1–6.

[3] D. Koller, G. Klinker, E. Rose, D. E. Breen, R. T. Whitaker, and M. Tuceryan, "Real-time vision-based camera tracking for augmented reality applications.," in *VRST*, 1997, vol. 97, pp. 87–94.

[4] A. Jalal, S. Kamal, and D. Kim, "A Depth Video-based Human Detection and Activity Recognition using Multi-features and Embedded Hidden Markov Models for Health Care Monitoring Systems.," *International Journal Interactive Multimedie Artificial Intelligence*, vol. 4, no. 4, 2017.

[5] A. Jalal, S. Kamal, and D. Kim, "Individual detection-tracking-recognition using depth activity images," in *2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2015, pp. 450–455.

[6] Z. Xu, L. Mei, C. Hu, and Y. Liu, "The big data analytics and applications of the surveillance system using video structured description technology," *Cluster Computing*, vol. 19, no. 3, pp. 1283–1292, 2016.

[7] Y.-L. Chen, T.-S. Chen, L.-C. Yin, T.-W. Huang, S.-Y. Wang, and T.-C. Chieuh, "City eyes: An unified computational framework for intelligent video surveillance in cloud environment," in *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, 2014, pp. 324–327.

[8] M. S. Hossain, M. M. Hassan, M. Al Qurishi, and A. Alghamdi, "Resource allocation for service composition in cloud-based video surveillance platform," in *2012 IEEE international conference on multimedia and expo workshops*, 2012, pp. 408–412.

[9] Q. Li, T. Zhang, and Y. Yu, "Using cloud computing to process intensive floating car data for urban traffic surveillance," *International Journal of Geographical Information Science*, vol. 25, no. 8, pp. 1303–1322, 2011.

[10] D. J. Neal and S. Rahman, "Video surveillance in the cloud?," *arXiv Prepr. arXiv1512.00070*, 2015.

[11] R. Chandana, S. Jilani, and S. J. Hussain, "Smart surveillance system using thing speak and Raspberry Pi," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 7, pp. 214–218, 2015.

[12] K. Abas, K. Obraczka, and L. Miller, "Solar-powered, wireless smart camera network: An IoT solution for outdoor video monitoring," *Computer Communications*, vol. 118, pp. 217–233, 2018.

[13] Y. Wang, C. von der Weth, T. Winkler, and M. Kankanhalli, "Tweeting Camera: A New Paradigm of Event-based Smart Sensing Device," in *Proceedings of the 10th International Conference on Distributed Smart Camera*, 2016, pp. 210–211.

[14] R. Du, S. Bista, and A. Varshney, "Video fields: fusing multiple surveillance videos into a dynamic virtual environment," in *Proceedings of the 21st International Conference on Web3D Technology*, 2016, pp. 165–172.

[15] T. Sieberth, R. Wackrow, and J. H. Chandler, "Automatic detection of blurred images in UAV image sets," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 122, pp. 1–16, 2016.

[16] M. S. Bakli, M. A. Sakr, and T. H. A. Soliman, "A spatiotemporal algebra in Hadoop for moving objects," *Geo-spatial Information Science*, vol. 21, no. 2, pp. 102–114, 2018.

[17] A. Jalal, M. A. K. Quaid, and K. Kim, "A Wrist Worn Acceleration Based Human Motion Analysis and Classification for Ambient Smart Home System," *Journal of Electrical*

Engineering and Technology, pp. 1–7, 2019.

[18] A. Jalal and M. Mahmood, "Students' behavior mining in e-learning environment using cognitive processes with information technologies," *Education and Information Technologies*, pp. 1–25, 2019.

[19] A. Jalal, M. A. K. Quaid, and M. A. Sidduqi, "A Triaxial acceleration-based human motion detection for ambient smart home system," in *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2019, pp. 353–358.

[20] A. Jalal and S. Kamal, "Improved Behavior Monitoring and Classification Using Cues Parameters Extraction from Camera Array Images.," *International Journal of Interactive Multimedia Artificial Intelligence*, vol. 5, no. 5, 2019.

[21] Q. Huang, J. Yang, and Y. Qiao, "Person re-identification across multi-camera system based on local descriptors," in *2012 Sixth International Conference on Distributed Smart Cameras (ICDSC)*, 2012, pp. 1–6.

[22] A. Jalal, Y. Kim, S. Kamal, A. Farooq, and D. Kim, "Human daily activity recognition with joints plus body features representation using Kinect sensor," in *2015 International Conference on Informatics, Electronics & Vision (ICIEV)*, 2015, pp. 1–6.

[23] H. Yoshimoto, N. Date, and S. Yonemoto, "Vision-based real-time motion capture system using multiple cameras," in *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI2003.*, 2003, pp. 247–251.

[24] A. Jalal and S. Kamal, "Real-time life logging via a depth silhouette-based human activity recognition system for smart home services," in *2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2014, pp. 74–80.

[25] A. Jalal and S. Kim, "The mechanism of edge detection using the block matching criteria for the motion estimation," *한국HCI 학회 학술대회*, pp. 484–489, 2005.

[26] A. Jalal and M. A. Zeb, "Security and QoS optimization for distributed real time environment," in *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*, 2007, pp. 369–374.

[27] D. Singh and C. K. Mohan, "Graph formulation of video activities for abnormal activity recognition," *Pattern Recognition*, vol. 65, pp. 265–272, 2017.

[28] K. Kim, A. Jalal, and M. Mahmood, "Vision-Based Human Activity Recognition System Using Depth Silhouettes: A Smart Home System for Monitoring the Residents," *Journal of Electrical Engineering and Technology*, pp. 1–7, 2019.

[29] M. Mahmood, A. Jalal, and M. A. Sidduqi, "Robust Spatio-Temporal Features for Human Interaction Recognition Via Artificial Neural Network," in *2018 International Conference on Frontiers of Information Technology (FIT)*, 2018, pp. 218–223.

[30] A. Jalal, S. Kamal, and C. A. Azurdia-Meza, "Depth maps-based human segmentation and action recognition using full-body plus body color cues via recognizer engine," *Journal of Electrical Engineering and Technology*, vol. 14, no. 1, pp. 455–461, 2019.

[31] Z. Shao, J. Cai, and Z. Wang, "Smart monitoring cameras driven intelligent processing to big surveillance video data," *IEEE Trans. Big Data*, vol. 4, no. 1, pp. 105–116, 2017.

[32] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Data processing and sharing for hybrid cloud-edge analytics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 9, pp. 2004–2017, 2018.

[33] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3539–3548.

[34] L. Beyer, S. Breuers, V. Kurin, and B. Leibe, "Towards a principled integration of multi-camera re-identification and tracking through optimal bayes filters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 29–38.

[35] C. Stahlschmidt, A. Gavriilidis, J. Velten, and A. Kummert, "Applications for a people detection and tracking algorithm using a time-of-flight camera," *Multimedia Tools and Applications*, vol. 75, no. 17, pp. 10769–10786, 2016.

[36] Y. T. Tesfaye, E. Zemene, A. Prati, M. Pelillo, and M. Shah, "Multi-target Tracking in Multiple Non-overlapping Cameras Using Fast-Constrained Dominant Sets," *International Journal of Computer Vision*, pp. 1–18, 2019.

[37] M. C. Liem and D. M. Gavrila, "Joint multi-person detection and tracking from overlapping cameras," *Computer Vision and Image Understanding*, vol. 128, pp. 36–50, 2014.

[38] V. P. Bhuvana, M. Schranz, C. S. Regazzoni, B. Rinner, A. M. Tonello, and M. Huemer, "Multi-camera object tracking using surprisal observations in visual sensor networks," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, p. 50, 2016.

[39] J. Wang, J. Pan, and F. Esposito, "Elastic urban video surveillance system using edge computing," in *Proceedings of the Workshop on Smart Internet of Things*, 2017, p. 7.

[40] N. Jiang, S. Bai, Y. Xu, C. Xing, Z. Zhou, and W. Wu, "Online inter-camera trajectory association exploiting person re-identification and camera topology," in *2018 ACM Multimedia Conference on Multimedia Conference*, 2018, pp. 1457–1465.

[41] K. A. Shiva Kumar, K. R. Ramakrishnan, and G. N. Rathna, "Distributed person of interest tracking in camera networks," in *Proceedings of the 11th International Conference on Distributed Smart Cameras*, 2017, pp. 131–137.

[42] T. Wang, J. Chen, P. Honeine, and H. Snoussi, "Abnormal event detection via multikernel learning for distributed camera networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 9, p. 989450, 2015.

[43] X. Wang, "Intelligent multi-camera video surveillance: A review," *Pattern Recognition. Lett.*, vol. 34, no. 1, pp. 3–19, 2013.

[44] L. Esterle, P. R. Lewis, R. McBride, and X. Yao, "The future of camera networks: Staying smart in a chaotic world," in *Proceedings of the 11th International Conference on Distributed Smart Cameras*, 2017, pp. 163–168.

[45] R. Verma and J. Ali, "A comparative study of various types of

image noise and efficient noise removal techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 10, 2013.

[46] H. Soleimani and S. Zafar, "Review - Moving object detection using background subtraction," 2018.

[47] H. Singh and J. S. Sodhi, "Image enhancement using sharpen filters," *International Journal of Latest Trends in Engineering and Technology*, vol. 2, no. 2, pp. 84–94, 2013.

[48] "VGG Face-16 CNN Model." .

[49] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," 2008.

[50] "CS231n Convolutional Neural Networks for Visual Recognition." .

[51] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis, "Human detection using partial least squares analysis," in *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, 2009, pp. 24–31.

[52] Y. Xing, H. Nagahashi, and X. Zhang, "A 3D Dynamic Visualization Surveillance System," *International Journal of Computer Science Issues*, vol. 13, no. 05, pp. 36–44, 2016.

[53] A. Laghaee, "BEHAVE Interactions Test Case Scenarios," 2007. .

[54] H. Wu, W. Pan, X. Xiong, and S. Xu, "Human activity recognition based on the combined svm&hmm," in *2014 IEEE International Conference on Information and Automation (ICIA)*, 2014, pp. 219–224.

[55] L. Piyathilaka and S. Kodagoda, "Gaussian mixture based HMM for human daily activity recognition using 3D skeleton features," in *2013 IEEE 8th conference on industrial electronics and applications (ICIEA)*, 2013, pp. 567–572.